LEPISME

François Boulier, François Lemaire, Lilianne Denis-Vidal, Thibaut Henin

LÉPISME ⋆

François Boulier [a] Lilianne Denis–Vidal [a]
Ghislaine Joly–Blanchard [b] François Lemaire [a]

[a] *Université Lille 1, 59655 Villeneuve d'Ascq France*
[b] *Université de Technologie de Compiègne*

Abstract

We present a first version of a software dedicated to an application of a classical nonlinear control theory problem to the study of compartmental models in biology. The software is being developed over a new free computer algebra library dedicated to differential and algebraic elimination.

Key words: differential elimination, compartmental models, biology, software.

1 Introduction

The acronym LÉPISME [1] (see the URL www.lil.fr/~lemaire/lepisme) stands for "*logiciel dédié à l'estimation de paramètres et à l'identification systématique de modèles*" which means, in French: software dedicated to parameters estimation and to systematic identification of models. A lepism is also a small insect, sometimes called "silver fish" that can be found in humid and dark places. Fining a lepism is also assumed to bring luck.

The software we present in this paper is dedicated to the parameters estimation problem in compartmental models, which are modelling tools quite used in biology (Cherruault, 1998). There are different issues. First issue: providing a tool permitting to practiotioners to prove that some of their models are false. Indeed, biological systems are very difficult to model: there are thirty

⋆ This is a full version of the article published in ICPSS 2004 proceedings.

Email addresses: `boulierlifl.fr` (François Boulier), `denvid@attglobal.net` (Lilianne Denis–Vidal), `Ghislaine.Joly-Blanchard@utc.fr` (Ghislaine Joly–Blanchard), `lemairelifl.fr` (François Lemaire).

[1] This work was partially funded by a French MathSTIC grant.

thousands of genes, hundreds of thousands of proteins in the case of human beings, how many possible interactions ? Tools able to take as input models, measures and prove that models are wrong are necessary. Our software is an attempt in that direction. Second issue: involving non trivial computer algebra methods in such an integrated software. Indeed, many computer algebra methods exist in computer algebra systems but there are basically never used in scientific software which are not primarily dedicated to computer algebra. Our software is built over a computer algebra library that we wrote in the C programming language (see the URL www.lifl.fr/~boulier/BLAD). Our goal while developing this library was to provide sort of an analogue, in the context of differential elimination, of the impressive GNU Multi Precision library, a library easy to plug in any software, dedicated to big numbers only. Third issue: hiding the technical aspects of the involved computer algebra methods (differential regular chains, differential elimination theory ...) which are actually impossible to understand by almost any researcher working outside our community. We believe that this legibility issue is crucial to develop the use of computer algebra in scientific computing.

Any compartmental model can be translated as a system of nonlinear, ordinary differential equations depending on paramaters. We restrict ourselves to polynomial systems, the other ones falling outside the scope of our methods. The problem we address can be stated as follows: given a polynomial parametric system of ODE and a set of measures (i.e. files of points $(t_i, x(t_i))$) for *some* of the variables (called *observed* variables), estimate the value of the parameters which fits the measures the best. There exist classical methods (Walter, 1994). They assume that approximate values for the parameters are known in advance. They improve then these initial values by means of optimisation methods (nonlinear least squares, which amount to Newton's method). Their drawback is obvious: they lead quite often to wong values of the parameters (local minima) even when the parameters values are theoretically uniquely determined by the measures (i.e. when the model is *globally observable*). The method we develop was validated a few years ago in (Noiret, 2000; Denis-Vidal et al., 2001, 2003). Stated in a nutshell, the idea is: by means of differential elimination (Boulier et al., 1995, 1997, 2001), the nonlinear least squares problem (which require the knowledge of an initial value) can be reformulated as a linear least square problem, for which no *a priori* knowledge is necessary. More precisely, by a mixed numeric (linear least squares) and symbolic (differential elimination) algorithm, one can automatically provide to the user an approximate guess of the parameters values which can then be used as a starting point by the classical optimisation method. Our method cannot guarantee that the provided starting point actually leads to the global minimum but this is already an improvement.

Our project presents an interesting feature: it is complementary to the existing methods; it does not replace them. Our software only relies on open

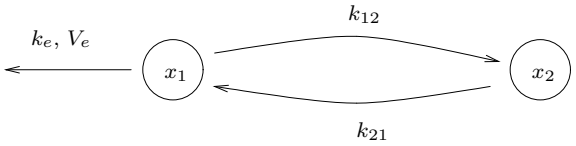
source software. It is actually protected by the GNU General Public License. The paper is organised as follows: we first recall the basics on compartmental models, identifiability and parameters estimation. Second we describe the mixed numeric and smbolic method that we apply. Last we describe the software, focusing on the computer algeba BLAD libraries.

2 Compartmental models, identifiability and parameters estimation

The problem states: given a compartmental model and a set of measures, estimate the values of the model parameters.

2.1 Compartmental models

The following compartmental model admits a pharmacokinetics interpretation. It describes the evolution of (say) a medical product between the blood (compartment 1) and some organ (compartment 2). The arrows denote exchanges between compartments: the product can go from each compartment to the other one. It can also exit from the blood by the action of kidneys.



To each compartment is associated a time varying variable: $x_i(t)$ denotes the amount of product present in compartment i at time t. In order to derive a system of differential equations from the model, one still needs to make some assumptions about the exchanges: it is assumed that the exchanges between the two compartments are linear and that the product exits from the blood by a Michaelis–Menten reaction. This being precised, the compartmental model is equivalent to a system of parametric ODE:

$$\dot{x}_1 = -k_{12}\,x_1 + k_{21}\,x_2 - \frac{V_e\,x_1}{k_e + x_1}, \quad \dot{x}_2 = k_{12}\,x_1 - k_{21}\,x_2.$$

In addition to the model, we assume we are given some measures. Here, we assume that $x_1(t)$ is known for $t = t_0, t_1, \ldots, t_N$ and that $x_2(t)$ is known to be zero at the origin: $x_2(t_0) = 0$. We may also make some assumptions on the model parameters k_{12}, k_{21}, k_e, V_e. Here, we assume k_e is known: $k_e = 7$.

To allow the reader to reproduce our results, we consider in this paper a file of

31 measures generated from $t_0 = 0$ to $t_{30} = 1.5$ with $k_{12} = 0.5$, $k_{21} = 3$, $V_e = 101$, $x_1(0) = 50$. We did not put any noise in our measures.

2.2 Identifiability and parameters estimation

A system identification based on physical laws often involves a parameter estimation. Before performing an estimation problem, it is necessary to investigate its identifiability. This a mathematical and a priori problem. We state it informally over our example: assume that the function $x_1(t)$ and all its derivatives of various orders are perfectly known ((e.g. error free) and well "behaved" (e.g. not identically zero), would the parameters of the model be uniquely determined? If the answer is yes, the model is said to be globally identifiable. If the model parameters may take a finite set of values then the model is said to be locally identifiable otherwise it is said to be unidentifiable.
The notion of identifiability has already been presented as an important notion in (Koopmans and Reiersol, 1950). But it was only in 1970 that Bellman and Åström (1970) have given formal basis for identifiability analysis of dynamical systems.
In the case of linear models several methods are available to analyse identifiability. The following approaches are readily used. One is based on the transfer function (Bellman and Åström, 1970), an other uses the Markov parameter matrix (Grewal and Glover, 1976), then the exhaustive modelling has been developed (Walter, 1994; Lecourtier, 1985).
In the case of nonlinear models these approaches cannot be used and other methods have been proposed. The linearization of the model has been considered by Grewal and Glover (1976). Some approaches are based on the Taylor series expansion (Pohjanpalo, 1978) or on generating power series (Walter and Lecourtier, 1982; Lecourtier, 1985; Lecourtier et al., 1987). The similarity transformation, based on the local isomorphism theorem, is another way to analyze identifiability of nonlinear controlled model (Vajda et al., 1989). It is an extension to the nonlinear case of the exhaustive modelling approach. More recently approches based on differential algebra have been proposed (Ollivier, 1997; Diop and Fliess, 1991; Ljung and Glad, 1994). Finally the identifiability question was recently addressed using probabilistic methods (Sedoglavic, 2002).
This investigation is the first step of the parameter estimation which is a practical question. It only makes sense if the model is at least locally identifiable. There exists a substantial litterature concerning the parameter estimation (Ljung, 1989). Generally the estimation methods are based on the choice of a criterion depending on the parameters and on the minimization of this criterion. The quadratic criteria are the most used. But few methods combine both identifibility and estimation (Ljung and Glad, 1994; Denis-Vidal et al., 2003). When the test of identifiability is done by differential elimination methods,

relations between parameters,inputs and outputs are obtained. Thus it should be interesting to use these relations in the parameter estimation.
In this contribution we propose an algorithm which links identifiability with numerical parameter estimation.
In the following we assume we deal with a compartment model which is at least locally identifiable. Our example is globally identifiable.

2.3 A numerical algorithm

Estimating parameters may be solved by means of purely numerical methods e.g. by nonlinear least squares. We have implemented this well known method using the Levenberg–Marquardt algorithm (Gill et al., 1988) which is a variant of the Newton method. We state it over our example.

Algorithm: optimize

(1) Assign random values to the parameters $\bar{k}_{12}$, $\bar{k}_{21}$, $\bar{V}_e$ (recall that k_e is known).
(2) Integrate numerically the differential system. This provides some values for $x_1(t)$ which are denoted $\bar{x}_1(t_0)$, $\bar{x}_1(t_1)$, ... , $\bar{x}_1(t_N)$.
(3) The criterion to minimize is $r = f_1^2 + \cdots + f_N^2$ where $f_i = x_1(t_i) - \bar{x}_1(t_i)$. Evaluate it. If the error is small enough stop computations else update the values of the parameters by the Levenberg–Marquardt method and go to step 2.

Remarks. The problem of such a method is well known: the algorithm may very well end up in a local minimum and miss the actual values of the parameters. Trying with $V_e = 90$, $k_{12} = 0.4$, $k_{21} = 0.01$ one ends up with $k_{12} = 0.77$, $k_{21} = 0.17$, $Ve = 82.82$ and a $3.\ 10^{-1}$ error.

The method may also fail (computations being interrupted by a "timeout" exception in our implementation) when the adaptive step–size numerical integrators of the Gnu Scientific Library enter difficult areas. Try with $V_e = 40$, $k_{12} = 0.4$, $k_{21} = 0.01$.

The method needs to integrate not only the two ODE system but also six extra ODE (the Fisher sensibility matrix) which give the sensitivities of $x_1(t)$ and $x_2(t)$ w.r.t. the variation of each parameter. These six ODE are symbolically generated by our BLAD libraries.

3 Guessing a good starting point

3.1 Overview

The numerical algorithm optimize presented above is sensitive to the values of the parameters you start with. It is proved in (Noiret, 2000; Denis-Vidal et al., 2003) that optimize can be greatly improved by guessing good initial values for the parameters using the following computer algebra method, based on differential elimination.

Let's assume that the right hand sides of the system equations are multivariate rational fractions.

Algorithm: guess

1. Differential elimination. By differential elimination methods, compute a set of differential polynomials which are consequences of the dynamical system and which only involve the measured variables, some of their derivatives of various orders and the unknown system parameters. Those differential polynomials, often called "input/output equations", have the form $c_1\,t_1 + \cdots + c_q\,t_q$ where the $t_1, \ldots, t_q$ are polynomials over the alphabet of the measured variables and their derivatives and the coefficients $c_1, \ldots, c_q$ are multivariate polynomials over the alphabet of the system parameters. We call them "blocks of parameters". Our example leads to only one input/output equation: the blocks of unknown parameters are enclosed between square brackets. They are multiplied by power products of the measured variables (x_1 and its derivatives) and the known parameter k_e:

$$\ddot{x}_1\,(x_1 + k_e)^2 + [k_{12} + k_{21}]\,\dot{x}_1\,(x_1 + k_e)^2 + [V_e]\,\dot{x}_1\,k_e + [k_{21}\,V_e]\,x_1\,(x_1 + k_e).$$

2. Estimating the block values. Using the measures, evaluate numerically the polynomials t_i for many different values of the time $t = t_0, \ldots, t_N$. This provides an overdetermined linear system of $N + 1$ equations whose unknowns are the blocks of parameters c_i. Solve this system using (say) the linear least squares method. This provides estimated values $\bar{c}_i$ for the c_i. Over our example, one gets

$$(k_{12} + k_{21}) = 2.1, \quad V_e = 87.29, \quad k_{21}\,V_e = 144.01.$$

Compare with the right values: $(k_{12} + k_{21}) = 3.5$, $V_e = 101$, $k_{21}\,V_e = 303$.

3. Estimating the parameters values. Form a polynomial system $c_i = \bar{c}_i$ for $1 \leq i \leq q$ (each c_i being replaced by its expression in the parameters and each $\bar{c}_i$ being approximated by a numerical value) and solve it (see subsection 3.4

for details). The solution of the polynomial system provides estimated values for the system parameters. Over our example, one gets

$$k_{12} = 0.45, \quad k_{21} = 1.65, \quad V_e = 87.29.$$

If one provides the above values to the *optimize* algorithm, one gets the right values with a 10^{-5} error.

3.2 Differential elimination

The differential method used in LÉPISME is PODI (Boulier et al., 2001) which is a variant of PARDI for ordinary differential equations. The "input/output" equations are obtained by computing a differential regular chain [2] of the initial system for a special ranking. We do not recall details of the underlying theory (the differential algebra (Ritt, 1950; Kolchin, 1973)) for reasons of brevity. We only explain how our algorithm works on our example.

Our example can be viewed as follows $C : \dot{x}_1 = -k_{12}\,x_1 + k_{21}\,x_2 - \frac{V_e\,x_1}{k_e+x_1}$, $\dot{x}_2 = k_{12}\,x_1 - k_{21}\,x_2$, $\dot{k}_{12} = \dot{k}_{21} = \dot{V}_E = \dot{k}_E = 0$ where the parameters are seen as constant functions of the time.

The system C is a differential regular chain of the differential ideal I that it defines w.r.t the *ranking* $\mathcal{R} : \cdots > \ddot{x}_1 > \ddot{x}_2 > \dot{x}_1 > \dot{x}_2 > x_1 > x_2 > \cdots > \dot{k}_{12} > \dot{k}_{21} > \dot{V}_E > \dot{k}_E > k_{12} > k_{21} > V_E > k_E$.

This means that C can be viewed as the following rewriting system: $\dot{x}_1 \to -k_{12}\,x_1 + k_{21}\,x_2 - \frac{V_e\,x_1}{k_e+x_1}$, $\dot{x}_2 \to k_{12}\,x_1 - k_{21}\,x_2$, $\dot{k}_{12} \to 0$, $\dot{k}_{21} \to 0$, $\dot{V}_E \to 0$, $\dot{k}_E \to 0$. Derivatives of the left hand sides of the rewriting rules can be rewritten by differentiating the right hand sides (for example the term $\ddot{x}1$ can be rewritten using the derivative of the first rule).

A normal form algorithm is described in (Boulier and Lemaire, 2000) (it is based on the Ritt pseudo reduction). Because C is a differential regular chain, we have the nice property $p \in I \iff \mathrm{NF}(p, C) = 0$

The whole idea to compute the "input/output" equations is to compute a differential regular chain $\overline{C}$ of I for a well chosen ranking $\overline{\mathcal{R}}$. On our example, it suffices to choose $\overline{\mathcal{R}} : \cdots > \dddot{x}_2 > \ddot{x}_2 > x_2 > \cdots > \dddot{x}_1 > \ddot{x}_1 > x_1 > \cdots > \dot{k}_{12} > \dot{k}_{21} > \dot{V}_E > \dot{k}_E > k_{12} > k_{21} > V_E > k_E$.

The input/output equation which only involves x_1, its derivatives and the parameters is a "smallest" polynomial of I w.r.t $\overline{\mathcal{R}}$. It must belong to $\overline{C}$.

[2] a differential regular chain is equivalent to a Ritt characteristic set

Although it is possible to compute $\overline{C}$ directly using a generic method like Rosenfeld–Gröbner (available in the MAPLE package Diffalg), it is more efficient to reuse the known differential regular chain C to guide the computations using the membership test provided by C: this is done by PODI. Moreover, PODI is written to handle prime ideals (which is the case on our example).

The set $\overline{C}$ is computed by converting the system C into $\overline{C}$. The set $\overline{C}$ is build incrementally by taking the equations in C one by one. PODI performs the following steps on our example:

- step 1: set $\overline{C} = \emptyset$
- step 2: pick an equation in C, say $\dot{x}_1 = -k_{12}\,x_1 + k_{21}\,x_2 - \frac{V_e\,x_1}{k_e+x_1}$. For the new ranking $\overline{\mathcal{R}}$, the leading variable of this equation is x_2. Writing the equation as a rewriting rule, set: $\overline{C} = \{x_2 \to \frac{1}{k_{21}}(\dot{x}_1 + k_{12}\,x_1 + \frac{V_e\,x_1}{k_e+x_1})\}$
- step 3: pick the equation $\dot{x}_2 = k_{12}\,x_1 - k_{21}\,x_2$. Before inserting it in $\overline{C}$, rewrite it using $\overline{C}$ yielding an equation with leading variable $\ddot{x}_1$. We now have:

$$\overline{C} = \left\{ \begin{array}{l} x_2 \to \frac{1}{k_{21}}(\dot{x}_1 + k_{12}\,x_1 + \frac{V_e\,x_1}{k_e+x_1}), \\ \ddot{x}_1 \to -(k_{12}+k_{21})\,\dot{x}_1 - \frac{V_e\,\dot{x}_1\,k_e}{(x_1+k_e)^2} - \frac{k_{21}\,V_e\,x_1}{(x_1+k_e)} \end{array} \right.$$

The algorithm PODI terminates for the equations are pairwise irreducible. At step 3, it got to make sure using the known chain C that k_{12} is nonzero divisor modulo I. Over this example, there are no purely algebraic simplifications to perform over the result. The second equation in $\overline{C}$ is precisely the "input/output" equation presented at the beginning of the section.

3.3 Estimating the blocks values

The difficulty comes from the fact that one needs to estimate the values of $\dot{x}_1$ and $\ddot{x}_1$ at t_0, t_1, ... , t_N and this cannot be done very precisely. Observe that one could work around the equation and transform it as an integral equation. This would improve the result but one cannot anyway completely evacuate the difficulty.

In our implementation we interpolate the values of x_1 and evaluate derivatives over the interpolating curves. We use the splines of degree 3 provided by the Gnu Scientific Library. Céline Noiret used interpolation polynomials of higher degrees.

Solving the system $c_i = \bar{c}_i$ leads to difficulties: the system can be over or under determined and involves only exact coefficients apart the $\bar{c}_i$. Several approaches are possible.

A numerical approach. One can directly solve the system with numerical methods (as Céline Noiret does with nonlinear least squares). However the obtained solution is only meaningful if the system is globally identifiable and if the numerical algorithm has not been stuck in a local minimum. Note that the local/global identifiability could be tested using probabilistic tests (Sedoglavic, 2002).

Symbolic solving. This is what we use in the current version of LÉPISME. It consists in symbolically solving w.r.t. to the parameters the system $c_i = b_i$ where the b_i are new indeterminates. We use the PALGIE algorithm. If the parameters are rational functions of the b_i's, the system is globally identifiable. If the parameters are implicit functions of the b_i's, the system is locally identifiable. Otherwise, the system is not identifiable.

This method is naive and can require extensive computations. It could be optimized using the following ideas. First, one can get rid of non identifiable systems by performing a probabilistic test over the model equations using (Sedoglavic, 2002). Then, the idea consists in symbolically solving the system $c_i = \bar{c}_i$ (replacing the $\bar{c}_i$ by rational numbers). However, a difficulty arises: there sometimes exist algebraic relations between the c_i that the $\bar{c}_i$ may not satisfy. By overcoming this difficulty, one could be reduced to the problem of solving a zerodimensional algebraic system. Advantage of this method: one gets all the possible values for the parameters.

Last, in our implementation of *guess*, when many different input/output equations are available, we first solve the simplest ones (the ones of lowest order) and rewrite the other ones using the obtained values. This turns out to provide more accurate results than solving all equations together.

4 The software

The software is decomposed in different layers. The lower layers may be used independently of the upper ones. It has been developed using the automake/autoconf system which makes it easy to test if some particular softwares or libraries (e.g. GB+RS, TRIADE (Moreno Maza, 2000), SCILAB, MATLAB ...) are available on the user's computer. It relies on the Gnu Sci-

entific Library for numerical computations and on the Gnu Multiple Precision library for big numbers. Today, the software is restricted to small globally identifiable uncontrolled models but this is going to change.

<table>
<tr><td colspan="2">LÉPISME interface: model solver</td><td>model editor</td><td>JAVA</td></tr>
<tr><td colspan="2">LÉPISME core methods: optimize, guess</td><td>JGraph</td><td>C and JAVA</td></tr>
<tr><td>BLAD</td><td>GSL</td><td></td><td>C</td></tr>
<tr><td>GMP</td><td></td><td></td><td>C</td></tr>
</table>

4.1 The LÉPISME graphical interface

The interface is made of two distinct applications: a *model editor* which permits to the user to enter the model graphically and a *model solver* which permits to launch the LÉPISME core methods: *optimize* and *guess*.

The main functionnality of the model editor is to graphically manipulate (creation/modification by mouse) compartmental models described by graphs: the user can easily enter a compartmental model in a graphical way, avoiding typing equations directly.

The model solver looks is a graphical interface permitting to the user to launch the LÉPISME core methods (recall that the identifiability methods are not yet implemented and do not even appear on the model solver). The main goal of this interface is to hide as much as possible to the user the technical considerations. For instance, the model equations, the computed regular differential chains, the blocks of parameters are never displayed.

Concerning the implementation, we have chosen to write the interface in JAVA. The reasons for this are the portability and the large builtin facilities to create graphical applications that JAVA provides. Moreover, compartmental models are nicely implemented using objects : any new type of exchange can be introduced by only coding a few new classes. The display and manipulation of the compartmental models are achieved using the graph manipulating library JGraph (see http://www.jgraph.com) [3] .

[3] JAVA itself does not provide such graph libraries

The algorithms *optimize* and *guess* are implemented as two executables in the C programming language. They take as input compartmental models described in a *model* file and a *data* file (this design is inspired from that of the AMPL (Fourer et al., 1993) mathematical programming software) generated by the graphical interface. A model file is a text files composed of sections describing the compartments, the exchanges, the parameters and the commands. A data file is a text file composed of one section containing the numerical values of the known parameters and compartments. By splitting the model and data files, one can consider different data files (i.e. different sets of measures) corresponding to the same model.

We are today working on changing the syntax of our files. We plan to switch to an SBML syntax. SBML is a variant of the XML data description language suited to biological models. It is an acronym standing for "Systems Biology Markup Language" (Hucka et al., 2004). The use of this standard would increase the interoperability of our software with other softwares dedicated to modelling biological systems. In particular, we could offer to the user the alternative software such as say, *Cell Designer* (see the URL www.systems-biology.org), in place of our model editor.

4.3 The BLAD libraries

The BLAD libraries (read "Bibliothèques Lilloises d'Algèbre Différentielle"), written in the C programming language are dedicated to differential elimination. Their first version was released in August 2004, by the Computer Algebra team of the university Lille 1 (see the URL www.lifl.fr/~boulier/BLAD). There are four libraries, the lower ones being independent of the upper ones. The following table gives the library names and some of their key features.

BAD	differential elimination methods: PARDI, Rosenfeld–Gröbner
BAP	multivariate polynomials over GMP
BAV	differential rankings, orderings
BA0	memory management, exception handling, parsers

The BAD library. The main data structure provided by the BAD library is a unified concept of "regular chain" which applies as well to the algebraic as to the differential setting. The concept of regular chain generalizes the one of "characteristic set". In the algebraic case, it was initiated in (Kalkbrener, 1993) and then much developed in the computer algebra team of Daniel Lazard

(Moreno Maza, 1997; Aubry et al., 1999; Aubry, 1999). The above definition was adapted to the differential setting in (Lemaire, 2002) under the name: "differential regular chain".

In the BAD library, a regular chain is defined by two sets of polynomials and two sets of properties. The two sets of polynomials are on the one hand the mathematical regular chain itself and on the other hand an heuristic set of polynomials which lie in the ideal defined by the chain and help processing reductions. This is indeed an idea borrowed from Faugère: do not forget polynomials which arise early in computations: they often turn out to simplify a lot reductions.

The two sets of properties are on the one hand a set of structural properties and on the other hand a set of desired properties. Structural properties are properties of the chain which cannot be changed or achieved algorithmically: does the chain define a differential or a nondifferential ideal ? is the ideal defined by the chain prime or not ? The desired properties are properties of the chain which can be changed or achieved algorithmically: is the chain primitive ? is it squarefree ? is it autoreduced ? (Aubry et al., 1999) is it strongly normalized (Boulier and Lemaire, 2000) ? is it coherent (Rosenfeld, 1959) ? There are relationships between these properties: if the ideal is differential then the chain must be squarefree; the coherence property only makes sense for systems of PDE.

The main implemented algorithms are the PARDI (Boulier et al., 2001) and the Rosenfeld–Gröbner (Boulier et al., 1995, 1997) simplifiers. The normal formal algorithm decsribed in (Boulier and Lemaire, 2000) is implemented too. A special care was given to the implementation of the Ritt reduction algorithm: There are different implementations which differ of the way polynomials are represented. In particular the implementation which seems the most efficient tries to keep polynomials factored (not necessarily completely) and to perform pseudoreductions factorwise. Indeed, after a few steps and because of the pseudoreduction algorithm, the simplifiers such as PARDI tend to produce polynomials which involve as factors powers of initials and separants of other polynomials used for simplification.

The BAP library. It primarily aims at implementing differential polynomials for the BAD library. It implements them as multivariate polynomials over (mainly) the ring of the integers. For instance the differential polynomial $\dot{x}-t\,x$ is viewed in the BAD library as an element of the differential polynomial ring $\mathbb{Q}(t)\{x\}$. It is viewed, in the BAP library, as a plain multivariate polynomial in $\mathbb{Z}[t,\, x,\, \dot{x}]$.

A special care was taken to implement the gcd of two multivariate polynomials over the ring of the integer numbers. It was implemented using modular and

ideal–adic methods as described in (Geddes et al., 1992) and is thus close to that of the MAPLE software. It is a very large and difficult algorithm which relies for instance on the factorization of multivariate polynomials to avoid the expression swell in the Hensel lifting (Zassenhaus, 1969) and which makes use of multivariate polynomials with coefficients in $\mathbb{Z}/p^k\mathbb{Z}$.

Since the BAP library polynomials are assumed to be involved in simplification processes of differential polynomials which involve many parameters, a special care was taken for implementing differentiation. Each parameter k is handled internally as a plain differential indeterminate (thus encoding a time varying function) and the dynamical system in consideration is enlarged with an extra rule $\dot{k} = 0$ to express the fact that it's value does not actually vary with the time. Without any further care, some expression swell would arise during differentiation: this operation would first generate monomials involving derivatives of the parameters; these monomials would afterwards be rewritten to zero. To avoid this behaviour, the differentiation algorithm receives as an extra argument a table of the variables whose derivatives are going to be reduced to zero in order not to generate the pointless monomials.

The representation of the polynomials is a variant of the so called distributed representation. During the design of the library, the following features were desired:

(1) to provide an easy access to the coefficients of the polynomials w.r.t. any subset of its variables,
(2) to permit some compression mechanism.

Here are some reasons which make the firt point important: the pseudoreduction is involved in many algorithms and it implies to access to the coefficients of a polynomial w.r.t. its leading variable; the key algorithms based on Hensel liftings need also to access to the coefficients of the polynomials w.r.t. some variable, usually chosen heuristically; many basic algorithms such as the multiplication of two polynomials P and Q are much more efficient if one can split the set of the variables into three sets (that which appear in P but not in Q, that which appear in Q and not in P and that which appear in both) and view P and Q as polynomials with coefficients in the ring of polynomials which depend on their common variables.

The second point was motivated by the size of some intermediate polynomials which already reached (even for tractable problems) hundreds of thousands of monomials.

We chose a variant of the distributed representation. In this variant, polynomials are defined as "pieces" of an underlying sorted linear combination of terms. The underlying linear combination is made of a dynamical array of numerical coefficients and a dynamical array of terms. Different representations of terms

are provided. For instance, terms may be stored in a hash table (equality test between terms gets very fast) or stored directly in the array, in a compressed way. Compression is achieved by keeping up to date, for each polynomial, a bound d on its degree w.r.t. each variable v. Then, in each term t, the degree $\deg(t, v)$ is stored on about $\log_2(d)$ bits.

A polynomial is either a full linear combination of terms or a "piece" of it. For instance, a coefficient of polynomial w.r.t. its leading variable is defined by a first monomial, a last monomial and the (leading) variable, which must be factored out from the terms of the linear combination in order to get the terms of the coefficient. The mechanism is more complicated to access to the coefficients of a polynomial w.r.t. a non leading variable: one makes use of an indirection array in order to provide the monomial which constitute the coefficient.

Of course, some iterators are provided to make it easy for algorithms to run over the coefficients of the polynomials.

The BAV library. It implements the variables over which polynomials are built. Variables may be derivatives of dependent variables, independent variables or mere constants. Many differential rankings (i.e. total orderings on the infinite set of the derivatives of the dependent variables) are implemented and more generally orderings which are not rankings (i.e. not compatible with the action of the derivations). These latter ones turn out to be very useful for implementing efficient versions of many algorithms on polynomials.

The BA0 library. It implements the low level mechanisms. In particular, it provides two memory management mechanisms: an implementation of the method described in (Faugère, 1998) which is used by Faugère and Rouillier in their software and a two stacks mechanism. Both mechanisms share the following feature: each function can only recover the memory that it used or the memory that the subfunctions it called used: a function cannot recover the memory wasted by its calling functions.

Because of this feature, the Faugère and Rouillier method is very efficient for iterative algorithms in which each loop performed in a given function needs a relatively small amount of memory: in this case, memory can just be wasted up to saturation and completely recovered in one operation. It seems less suitable for very recursive methods (such as triangular sets ones) where memory must be recovered much more regularly. The two stacks mechanism provides then a simple and quite efficient alternative.

The library provides also an exception handling mechanism which permits to stop gracefully computations which exceed some given bounds in time or in memory. This mechanism is also used within the BLAD library. It was quite easy to design because of the carried out memory management mechanisms.

Indeed, the only difficulty arising when implementing such mechanisms consists in recovering the memory used between the moment where the exception catching point was set and the moment where the exception was thrown.

The library provides powerful parsers which turn out to be very interesting for performing esily some data type conversions. Such conversions are very rare within the BLAD library but very common in the LÉPISME core methods. Big numbers are handled by GMP.

5 Conclusion

Symbolic methods are usually very difficult to understand by practitioners (specialists spend years studying them). For this reason, we believe that it is very important to develop complete softwares (up to the graphical interface) in order to prove the usefulness of symbolic methods. For the same reason, usual computer scientists will never be able to understand our methods accurately enough to implement them: research papers often do not even mention some very difficult and necessary subalgorithms (e.g. the multivariate polynomials gcd used to factor out contents from equations). It is thus our task to implement the complete softwares.

This large work was strongly motivated by many discussions that the first author had with members of the computer algebra team of Daniel Lazard a few years ago.

References

Aubry, P., 1999. Ensembles triangulaires de polynômes et résolution de systèmes algébriques. Implantation en Axiom. Ph.D. thesis, Université Paris VI.

Aubry, P., Lazard, D., Moreno Maza, M., 1999. On the theories of triangular sets. Journal of Symbolic Computation 28, 105–124.

Bellman, R., Åström, K. J., 1970. On structural identifiability. Mathematical biosciences 7, 329–339.

Boulier, F., Lazard, D., Ollivier, F., Petitot, M., 1995. Representation for the radical of a finitely generated differential ideal. In: proceedings of ISSAC'95. Montréal, Canada, pp. 158–166.

Boulier, F., Lazard, D., Ollivier, F., Petitot, M., 1997. Computing representations for radicals of finitely generated differential ideals. Tech. rep., Université Lille I, LIFL, 59655, Villeneuve d'Ascq, France, (ref. IT306, december 1998 version published in the habilitation thesis of Michel Petitot).

Boulier, F., Lemaire, F., 2000. Computing canonical representatives of regular differential ideals. In: proceedings of ISSAC 2000. St Andrews, Scotland, pp. 37–46.

Boulier, F., Lemaire, F., Moreno Maza, M., 2001. PARDI ! In: proceedings of ISSAC'01. London, Ontario, Canada, pp. 38–47.

Cherruault, Y., 1998. Modèles et méthodes mathématiques pour les sciences du vivant. PUF, Paris.

Denis-Vidal, L., Joly-Blanchard, G., Noiret, C., 2001. Some effective approaches to check the identifiability of uncontrolled nonlinear systems. Math. Comp. in Simulation 57, 35–44.

Denis-Vidal, L., Joly-Blanchard, G., Noiret, C., 2003. System identifiability (symbolic computation) and parameter estimation (numerical computation). In: Numerical Algorithms. Vol. 34. pp. 282–292.

Diop, S., Fliess, M., 1991. Nonlinear observability, identifiability, and persistent trajectories. In: Proc. 30th CDC. Brighton, pp. 714–719.

Faugère, J.-C., january 1998. A fast, easy to use algorithm for dynamic memory management. Tech. rep., LIP6, université Paris VI, (unpublished).

Fourer, R., Gay, D. M., Kernighan, B. W., 1993. AMPL A Modeling Language For Mathematical Programming. International Thomson Publishing.

Geddes, K. O., Czapor, S. R., Labahn, G., 1992. Algorithms for Computer Algebra. Kluwer Academic Publishers.

Gill, P. R., Murray, W., Wright, M. H., 1988. The Levenberg–Marquardt Method. Wiley, New York.

Grewal, M. S., Glover, K., 1976. Identifiability of linear and non-linear dynamical systems. In: IEEE Trans. Autom. Control. Vol. AC–21. pp. 833–837.

Hucka, M., Finney, A., Bornstein, B., , Keating, S., Shapiro, B., Matthews, J., Kovitz, B., Schilstra, M., Funahashi, A., Doyle, J., Kitano, H., 2004. Evolving a lingua franca and associated software infrastructure for computational systems biology: theSystemsBiologyMarkupLanguage(SBML) project. IEE Systems Biology 1, 41–53.

Kalkbrener, M., 1993. A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties. Journal of Symbolic Computation 15, 143–167.

Kolchin, E. R., 1973. Differential Algebra and Algebraic Groups. Academic Press, New York.

Koopmans, T. C., Reiersol, O., 1950. The identification of structural characteristics. Ann. Math. Statist. 21 (2), 165–181.

Lecourtier, Y., 1985. Propriétés structurelles de modèles : études théoriques, tests formels, application à la catalyse hétérogène. Ph.D. thesis, Université Paris-Sud.

Lecourtier, Y., Lamnabhi-Lagarrigue, F., Walter, E., 1987. olterra and generating power series approaches to identifiability testing. Identifiability of Parametric Models, 75–84.

Lemaire, F., january 2002. Contribution à l'algorithmique en algèbre différentielle. Ph.D. thesis, Université Lille I, 59655, Villeneuve d'Ascq, France, in

French.

Ljung, L., 1989. System identification: practice for the user. Prentice–Hall, Englewood Cliffs.

Ljung, L., Glad, S. T., 1994. On global identifiability for arbitrary model parametrisations. Automatica 30, 265–276.

Moreno Maza, M., 1997. Calculs de Pgcd au–dessus des Tours d'Extensions Simples et Résolution des Systèmes d'Équations Algébriques. Ph.D. thesis, Université Paris VI, France.

Moreno Maza, M., 2000. On Triangular Decompositions of Algebraic Varieties. Tech. rep., NAG, (presented at the MEGA2000 conference).

Noiret, C., 2000. Utilisation du calcul formel pour l'identifiabilité de modèles paramétriques et nouveaux algorithmes en estimation de paramètres. Ph.D. thesis, Université de Technologie de Compiègne.

Ollivier, F., 1997. Identifiabilité des systèmes. Tech. rep., GAGE, École Polytechnique.

Pohjanpalo, H., 1978. System Identifiability based on the power series expansion of the solution. Math. Biosci 41, 21–33.

Ritt, J. F., 1950. Differential Algebra. Dover Publications Inc., New York.

Rosenfeld, A., 1959. Specializations in differential algebra. Trans. Amer. Math. Soc. 90, 394–407.

Sedoglavic, A., 2002. A Probabilistic Algorithm to Test Local Algebraic Observability in Polynomial Time. Journal of Symbolic Computation 33 (5), 735–755.

Vajda, S., Rabitz, H., Godfrey, K. R., 1989. Similarity transformation approach to identifiability analysis of nonlinear compartimental models. Math. Biosciences 93, 217–248.

Walter, É., 1994. Identifiability of State Space Models. Vol. 46 of Lecture Notes in Biomathematics. Springer Verlag.

Walter, E., Lecourtier, Y., 1982. Global approaches to identifiability testing for linear and non linear state space models. Math. Comput. Simulation 24, 472–482.

Zassenhaus, H. J., 1969. On Hensel Factorization I. Journal of Number Theory 1, 291–311.